(12) UK Patent Application (19) GB (11) 2 009 980 A

(21) Application No. 7847708
(22) Date of filing 8 Dec 1978
(23) Claims filed 8 Dec 1978
(30) Priority data
(31) 7713708
(32) 12 Dec 1977
(33) Netherlands (NL)
(43) Application published 20 Jun 1979
(51) INT CL[2] G06F 5/06 G11C 19/00
(52) Domestic classification G4A 13E 8C MT1
(56) Documents cited GB 1472210 GB 1412046
(58) Field of search G4A

(71) Applicants
N.V. Philips' Gloeilampenfabrieken
Emmasingel 29, Eindhoven, The Netherlands.

(72) Inventors
Pierre Gerardus Jansen
Jozef Laurentius Wilhelmus Kessels
Benny Louisa Angelina Waumas

(74) Agents
R. J. Boxall

(54) Data buffer memory

(57) A data buffer memory of the "first-in, first-out" type comprises a plurality of registers REG(0)...REG(n-1), a fixed input bus (IN) *via* which data are applied to the first register [REG(0)] and an output bus (OUTB) *via* which data are extracted from the buffer memory. The buffer memory comprises logic means [LM(0)...LM(n-1)] whereby a variable output register can be selected. The logic means [(LM(0)...LM(n-1)] provides, preferably by means of status signals [(s(i) and $\overline{s(i)}$)] indicating whether each register [REG (i)] is full or empty, in cooperation with write request and acknowledge signals [creq. ers] applied from outside the buffer memory, a data read-out signal [selout(i)], whereby data are read from the last full register, and a data shift signal (sh) when data in the buffer memory are to be shifted further from the input location as additional data are written in register [REG(0)]. The buffer memory may be made from integrated circuits.

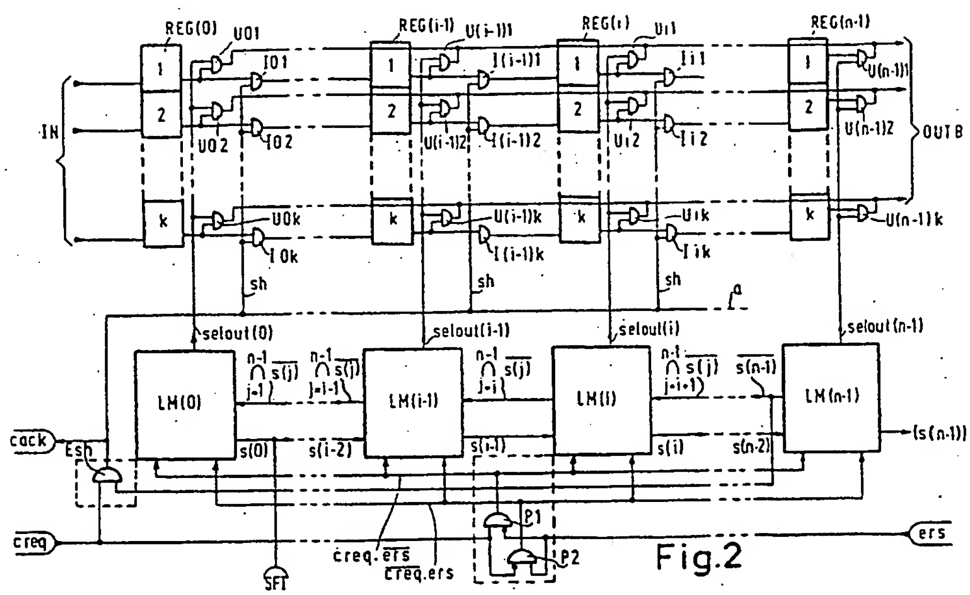

Fig.2

GB 2 009 980 A

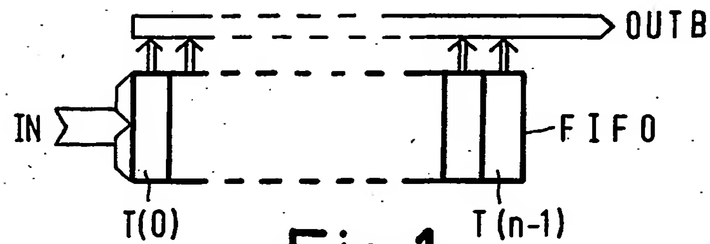

Fig.1

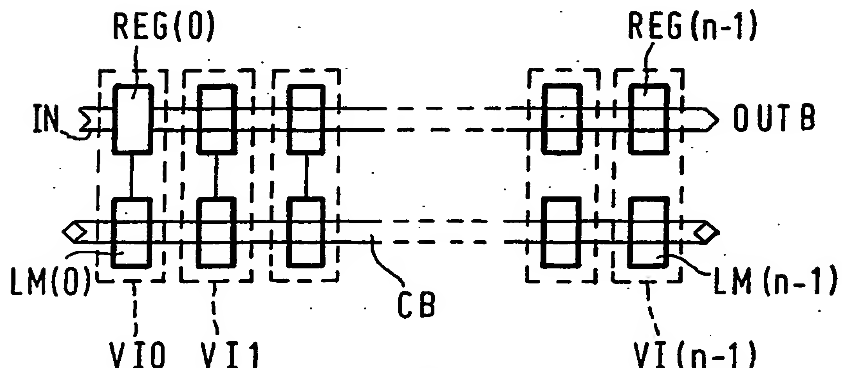

Fig.3

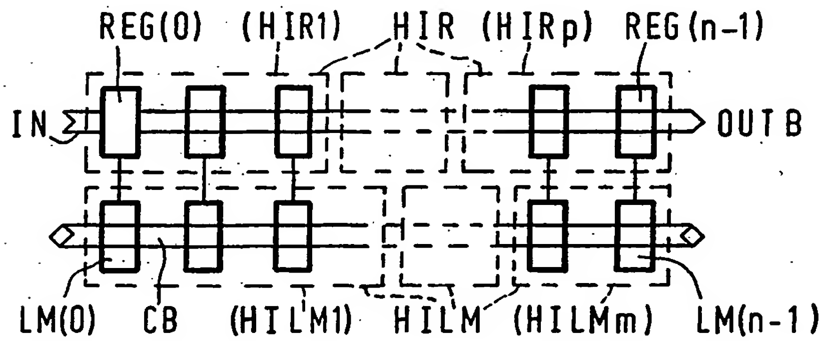

Fig.4

2/3

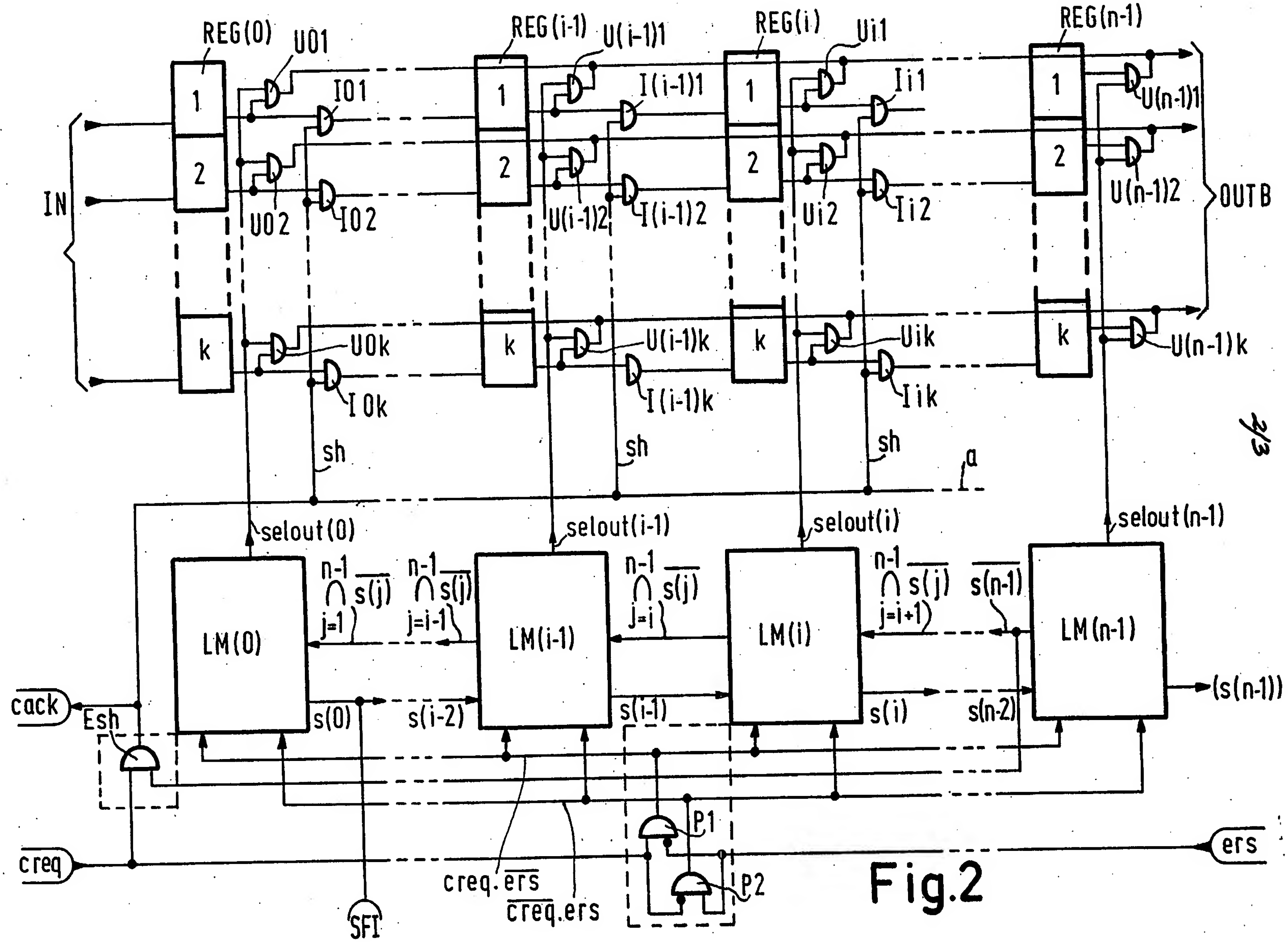
IN
REG(0)
U01
I01
U02
I02
U0k
I0k
REG(i-1)
U(i-1)1
I(i-1)1
U(i-1)2
I(i-1)2
U(i-1)k
I(i-1)k
REG(i)
Ui1
Ii1
Ui2
Ii2
Uik
Iik
REG(n-1)
U(n-1)1
U(n-1)2
U(n-1)k
OUTB
1
2
k
sh
a
selout(0)
selout(i-1)
selout(i)
selout(n-1)
LM(0)
LM(i-1)
LM(i)
LM(n-1)
$\bigcap_{j=1}^{n-1}\overline{s(j)}$
$\bigcap_{j=i-1}^{n-1}\overline{s(j)}$
$\bigcap_{j=i}^{n-1}\overline{s(j)}$
$\bigcap_{j=i+1}^{n-1}\overline{s(j)}$
$\overline{s(n-1)}$
s(0)
s(i-2)
s(i-1)
s(i)
s(n-2)
(s(n-1))
cack
Esh
creq
SFI
$creq.\overline{ers}$
$\overline{creq}.ers$
P1
P2
ers

Fig.2

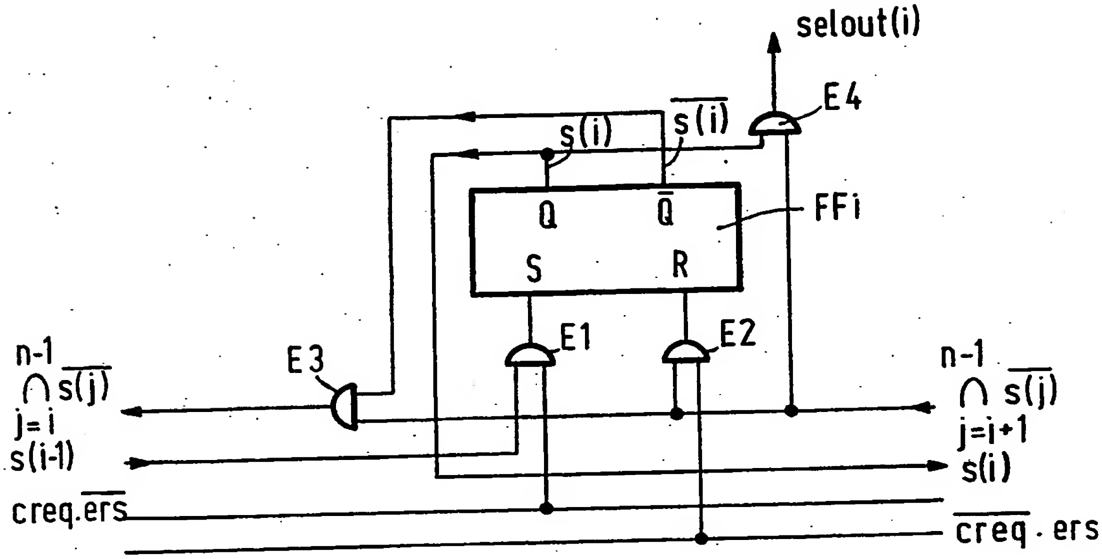
selout(i)
E4
$\overline{s(i)}$
s(i)
Q
$\overline{Q}$
FFi
S
R
E1
E2
E3
$\bigcap_{j=i}^{n-1} \overline{s(j)}$
s(i-1)
creq.$\overline{ers}$
$\bigcap_{j=i+1}^{n-1} \overline{s(j)}$
s(i)
$\overline{creq}$. ers

Fig.5

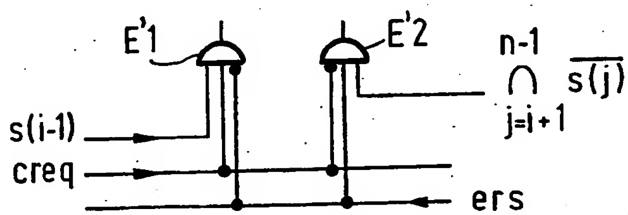
E'1
E'2
$\bigcap_{j=i+1}^{n-1} \overline{s(j)}$
s(i-1)
creq
ers

Fig.6

SPECIFICATION

Data buffer memory

The invention relates to a data buffer memory of the "first-in, first-out" type.

A wide variety of data buffer memories of the described "first-in, first-out" type are known; they are *inter alia* used as a buffer device in digital data processing and communication systems at areas where differences occur between data input and data output. A number of the known buffers are characterised by a simple construction, notably by a pronounced repetitive character of the various sections of the buffer. An example in this respect is formed by the buffer described in British Patent Specification No. 1,363,707. Buffers of this kind involve a problem in that, when the capacity of the buffer amounts to *n* sections, a message which is applied to an empty buffer appears on the output thereof only after *n* clock pulse cycles. Particularly if *n* is large (>32. . .) inadmissible delays are liable to occur in practice. These buffers are thus characterised as having a fixed input and a fixed output.

Also known are buffers which do not involve such a delay, because counting devices are used to activate a variable input location as well as a variable output location of the buffer. In such cases it is not necessary to transport the data each time through the entire buffer for transfer from an input to an output. Particularly in the case of an empty or almost empty situation, delays are thus prevented. Buffer devices of this kind are known from British Patent Specification No. 1479774. These buffer devices, however, involve a major problem in that the complexity of control substantially increases particularly in the case of buffers comprising a large number of sections. Counters having a high counting capacity and complex decoding selection networks for the inputs and outputs to be assigned or other additional steps are then required. Moreover, the linking of a number of small buffers in order to form one large buffer is not possible without giving rise to additional complications.

The invention has for its object to provide a buffer memory of the described type which has a simple construction and which, moreover, involves a short data delay time.

According to the present invention there is provided a data buffer memory of the "first-in, first-out" type, comprising a plurality of registers arranged in a series array, an input which is connected to a first of the registers and *via* which data to be written are introduced, an output bus coupled to all the registers *via* which bus data are read from th buffer, and logic means for ensuring that data written into the buffer is advanced register by register from the first of the registers and for assigning the register from which data is to be read-out.

The buffer memory thus obtained may be described as a buffer comprising a fixed input and variable output. Because of the variable output location, being each time situated as near as possible to the input in order to ensure a substantially uninterrupted content of the buffer, a minimum delay time of the buffer is obtained.

In a further embodiment of the buffer memory in accordance with the invention there are *n* sections (0,1. . .,n-1), each section containing one of the registers, the logic means being arranged so that in operation the following signals can be generated a) selout (i) = $s(i)\prod_{j=i+1}^{n-1}\overline{s(j)}$ which, if this condition is satisfied, represents the signal whereby the register (i) being the first filled register, viewed from the last section of the buffer, can be selected and hence connected to the output bus; s(i) indicating that the register (i) contains data and $\prod_{j=i+1}^{n-1}\overline{s(j)}$ indicating that the register downstream of the register (i) are empty;

b) sh = $\overline{s(n-1)}$.creq. which, if this condition is satisfied, is the shift signal for shifting the entire content of the buffer over one section in reaction to the appearance of a request signal (creq) which originates from outside the buffer and which indicates that data are applied to the input, this request being granted if at least the last section of the buffer is empty; this is indicated by a status signal $\overline{s(n-1)}$=1 of the latter section, register (n-1). For generating a status signal, the logic means are operable such that s(i): = 1 which, if the condition creq.$\overline{\text{ers}}$. s (i-1) is satisfied, is the status signal indicating the filling of a register (i) as a result of a request signal (creq) whereby the register (0) is filled with data from outside the buffer or a register (i) 1 ≠ 0 is filled with data in reaction to a shift signal whilst none of these registers has been emptied at the same time *via* the output bus, denoted by an acknowledge signal (ers) from outside the register, and at least the preceding register (i-1), with the exception of i = 0 has been filled; the status signal s(i):=0 indicates that the register (i) has been emptied *via* the output bus and has not been filled again at the sam time, at least the next register (i+1) being empty (written as $\overline{s(i)}=\overline{\text{creq}}$.ers. $\overline{s(i+1)}$). Linking of a plurality of buffers can thus be performed without any complications. In another embodiment of the buffer memory for the formation of a status signal s(i):=1 with the logic means, creq.$\overline{\text{ers}}$.$\prod_{j=0}^{i-1}$s(j) has to be satisfied which means that when a write request appears on the input, register (0), or the shifting in a register (i) respectively and the fact that the register (i+1) is not read at the same time (ers), all registers preceding the register (i) have been filled and that the status signal s(i):=0 if the condition $\overline{\text{creq}}$.ers.$\prod_{j=i+1}^{n-1}\overline{s(j)}$ has been satisfied; this occurs in the absence of a request for writing (creq) and when the register (i) is being emptied (ers) and subject to the condition that all subsequent registers are empty. In addition to the described properties, this embodiment of the buffer has a self-stabilising character. This means that there can be no situations in which an error, for example caused by a fault, can give rise to a permanent error situation. This is *inter alia* due to the fact that at no time doubt can arise as regards the location in the buffer where the register is to be connected to the output bus. The register (i), being the first filled register viewed from the last section of the buffer, is unambiguously determined.

Furthermore, in order to minimise the risk of data

loss due to any error occurring (not causing instability) a further embodiment of the buffer memory is characterised in that with the logic means for forming a status signal ($s(i){:}=1$) it applied when the condition creq.$\overline{ers}$.$s(i\text{-}1)$ is satisfied which is the cast upon appearance of a request for writing (creq) and the non-simultaneous reading (ers) of the preceding filled register (i-1) (except for i-0), whilst for the status signal $s(i){:}=0$ is applies that the condition $\overline{creq}.ers.\prod_{j=i+1}^{n-1}\overline{s(j)}$ is satisfied, which is the case in the absence of a request for writing ($\overline{creq}$) and during the emptying (ers) of the register (i) and subject to the condition that all subsequent registers are empty.

As a result of the use of said status per section, preferably being updated in bistable elements as part of the logic means, a simple arrangement is obtained which is suitable for integration purposes. Due to the modular construction, the relevant register and the associated logic means can be constructed as a solid state integrated circuit at least per section of the data buffer memory. It is alternatively possible for the buffer to consist of at least one group of registers and at least one group of logic means per section of the buffer, said group being solid state integrated circuits. The modular construction also implies that a plurality of buffer memories can be readily connected one behind the other in order to realise buffer lengths as desired.

The present invention will now be described, by way of example, with reference to the accompanying drawings, wherein:

Figure 1 shows a circuit diagram of a "first-in, first-out" buffer memory in accordance with the invention, comprising a fixed input and a variable output,

Figure 2 shows a more detailed block diagram of the buffer memory in accordance with the invention;

Figures 3 and *4* show examples of the partitioning of the buffer memory in view of construction in the form of solid state integrated circuits;

Figure 5 shows an example of the logic means of a section of the buffer memory; and

Figure 6 shows a detail of Figure 2, together with a detail of Figure 5, in a slightly different embodiment.

Figure 1 shows a simplified diagram of a "first-in, first-out" buffer memory FIFO comprising a fixed input and a variable output. A fixed input IN is situated at the input of the first section T(0) of the buffer memory. OUTB forms an output bus *via* which data are extracted from the buffer, notably from outputs of an assigned register section T(0), T(1), . . . T(n-1) thereof.

Figure 2 shows a more detailed block diagram as an example of the construction of the buffer of Figure 1 notably in a modular arrangement. The buff r consists of a register section comprising the registers REG(0),. . . REG(i-1),. . . REG(n-1). These registers serve for the storage of applied data. Each register may consist of one or more stages 1, 2, . . . *k*. It is thus indicated that the data path can be chosen at random as far as the width is concerned. For each bit of width of the data path, a stage 1, 2, . . . *k* per REG(i) is required. The regist r REG(0) of the first section of the buffer memory serves as the input register for the entire buffer. The input IN (an input terminal for each bit of the data path) is connected to REG(0). The output bus OUTB is shown to extend across the registers in Figure 2. Each register REG(i) has its outputs (of each stage 1, 2, . . . *k*) connected to the bus OUTB. For this purpose, use is made of AND-function gates: UO1, UO2, . . . UOk for the relevant register stages 1, 2, . . . *k* of REG(0); Ui1, Ui2, . . . Uik for the relevant register stages 1, 2, . . . *k* of REG(i) etc. The selection as regards which of the register REG(i) where i=0, . . . (n-1), is connected to the output bus OUTB is determined by the logic means LM(0),. . . LM(i-1), LM(i) . . . LM(n-1) which are provided per section of the buffer. A signal selout (0), . . . selout (i), . . . or selout (n-1) is generated in the logic means and is applied to said AND-function gates U01, . . . UO*k* . . . or U*i*1 . . . U*ik* . . . or U(n-1)1, . . . U(n-1)k. Thus, the selection is effected of the one register REG(i) where i = 0 . . . (n-1) wherefrom data are applied to the output bus OUTB of the buffer.

For the shifting of the data between the sections inside the buffer, occurring when new data are applied to the buffer and space is still available in the buffer, connections are provided between the stages of the various registers, said connections extending *via* AND-function gates I01, I02 . . . I0*k* . . . and I*i*1, I*i*2 . . . I*ik*, respectively between an output of a given register stage and an input of a register stage of the next section of the buffer. In the embodiment shown in Figure 2, the output for each register stage are shown in common, i.e. for the output to the output bus OUTB as well as the output to the input of a register stage of the next section of the buffer. This shifting takes place under the control of the shift signals "sh" which are generated in the logic means LM(i).

The buffer memory furthermore consists of a control section comprising a logic means LM(i) where i=0 . . . (n-1) per section of the buffer. The signals generated in these logic means are, in addition to the said signal selout(i) the status signals s(i) and s(i) which form an indication "full" and "empty", respectively, of a register (i) and also a preferably used combinatory form thereof: $\prod_{j=i+1}^{n-1}\overline{s(j)}$; this means that on the basis of the Boolean AND-function of all registers REG (i+1)up to and including REG(n-1) the status signal s(j) has a value zero (which means $\overline{s(j)}$ = 1). This is the definition of the condition that all registers succeeding REG(i) are empty. The indication "empty" means that no valid data are present therein. Further particulars in this respect will be given with reference to Figure 5. In the control section an AND-function gate P1 monitors a condition creq.$\overline{ers}$. in this example. This means that each time when this condition is satisfied, a signal having the logic value "1" appears on the output of P1. This signal is applied to all logic means LM(i) (line denoted by creq.$\overline{ers}$). The signal "creq" represents a request from outside the buffer for transferring data to the buffer. The signal "ers" represents an acknowledge signal from outside the buffer which occurs (logic 1=value) when data

have been taken over from the buffer. Thus creq.$\overline{ers}$. is fulfilled if there is a request to write-in data and there being nonread-out data acknowledgement. Similarly, in this example, the inverse condition creq.$\overline{ers}$. is monitored in an AND-function gate P2. This condition is satisfied if the request signal "creq" does not appear simultaneously with the acknowledge signal "ers". Thus $\overline{creq}$.ers. is fulfilled if there is a read-out data acknowledgement and at the same time there being no request to write-in data. This signal "creq" is also applied to all logic means . . . LM(i) . . .. In an AND-function gate Esh, the said signal 'sh" is generated. Each time when the request signal "creq" appears and the buffer has not been completely filled yet, denoted by the status $\overline{s(n-1)}$=1, the gate Esh supplies the shift signal "sh" which is applied to each register of all sections of the buffer, except for the last register. This can be expressed in a formula as follows: sh = creq.$\overline{s.(n-1)}$. When this expression is satisfied, the shift operation will take place in reaction to a clock signal (not shown), which means that the clock signal is subject to the condition that "sh" occurs. This signal "sh" can also be used if desired, as the acknowledge signal "cack" in order to indicate that the request "creq" has been reacted to, which means that the data applied to the input IN have indeed been stored in the first section (REG(0)). The complete data content of the buffer are thus shifted over one section as one block. The gate Esh is assumed to be included in the logic means LM(0). Similarly, the gates P1 and P2 are assumed to be included in arbitrary logic means LM(i). Finally Figure 2 shows a line which is denoted by the reference SF1 and which extends outside the buffer. SFI = s(0), which means that SFI has a logic 1-value as long as s(0)=1, i.e. as long as REG(0) contains data. This is an indication that the buffer still contains data, so that data are present on the lines OUTB.

Figures 3 and 4 show a number of possibilities for partitioning the buffer memory in view of construction in the form of solid state integrated circuits (ICs). The modularity of the buffer memory shown in Figure 2 enables a variety of solutions: the references VI0, VI1, ... VI(n-1) in Figure 3 indicate that integration is possible at least per section of the buffer: a REG(0) is combined in an IC with logic means LM(0). The connection between all sections VIi is formed in the register section (upper part of the drawing of Figure 3) by the output bus OUTB, and so are the connections within the bus OUTB in the drawing between the stages of the successive buffer sections for the shifting of the data from one section to a next section. The connections between the logic means and the further input and output signals are denoted in Figure 3 by a signal line bundle CB. Similarly, it is indicated in Figur 4 that integration in integrated circuits is possible per group HIR or groups HIR1 . . . HIRp of registers REG(0) . . . REG(n-1) or per group HILM or groups HILM1 . . . HILMm of logic means LM(0) . . . LM(n-1).

Obviously, the construction of complete buffer memories in one solid state integrated circuit is also possible. The linking of the sections or complete buffer memories does not impose problems, as will yet be described with reference t Figure 5.

Figure 5 shows an embodiment of the logic means LM)i) of a buffer section (i) for controlling the register REG(i) of this section. The logic means in this embodiment comprise a flip-flop FFi, having a set input S and a reset input R and outputs Q and $\overline{Q}$. Also shown are four logic AND-function gates, E1, E2, E3 and E4. The construction of the logic means is simple and offers proper operation of the buffer memory when the memory elements (flip-flops) used like the memory cells of the registers, are capable of reading themselves; to this end, use can be made of, for example, edge-triggered D-types. These elements are commercially available (for example, type indication 74LS74). In other embodiments other logic elements such as NAND-gates etc. can alternatively be used within the scope of the invention. For the memory elements use can notably be made also of flip-flops of the master-slave type. The essential aspect is that the logic functions to be performed by the logic means can indeed be realised. When use is made of master-slave flip-flops, at least two clock pulse signals will have to be used instead of one clock pulse signal. In order to give an idea of the implications thereof, reference is made to the copending Patent Application No. 47706/78 filed simultaneously with the present Application and claiming priority from Netherlands Patent Application No. 77 13706.

The functions realised in the logic means are such that the desired signals for the control of the buffer are generated. These signals are:

a) selout(i) which is the signal which provides the selection of the register REG(i) wherefrom data are read to the output bus OUTB. This signal selout(i) = 1 appears if the condition s(i). $\bigwedge_{j=i+1}^{n-1} \overline{s(j)}$ is "true" (which means that it has the logic value "1"). In the AND-function gate E4, it is determined whether this condition is satisfied. The status s(i) of the relevant section is then considered: the section must be filled, so s(i)=1. Furthermore, all further sections downstream of the relevant section of the buffer must be empty. This is determined by the expression $\bigwedge_{j=i+1}^{n-1}$ s(j). This takes place in the preceding section logic means LM(i+1) or LM(i) for LM(i-1). This is realised in the AND-function gate E3: therein it is determined whether $\bigwedge_{j=i}^{n-1} \overline{s(j)}$ = 1, i.e. whether the condition is satisfied that the register REG(i) and all subsequent registers (which explains the symbol $\bigwedge$ as the Boolean AND-function symbol) are empty (statuses s(i'=0).

b) sh=$\overline{s(n-1)}$.creq. which is the shift signal which is formed in the gate Esh already described with reference to Figure 2. This gate Esh and also the gates P1 and P2 mentioned with reference to Figure 2 may be accommodated in one of the logic means. This is indicated in Figure 2 by the inclusion of P1 and P2 in LMi and Esh in LM(0) (see broken lines). In the case of construction in the form of integrated circuits, said gates may occur a number of times (in order to maintain repetitiveness), and are then connected to signals creq and ers only in as far as is necessary for obtaining the desired signals "sh", creq.$\overline{ers}$. and $\overline{creq}$.ers. See also the description

given with reference to Figure 6, indicating how the gates P1 and P2 may be included in the gates E1 and E2 per LM(i).

c) status signals s(i) (and $\overline{s(i)}$). In order to set and reset the statuses *via* the inputs S and R of FFi, a number of possibilities exist;

the first possibility is: $set(i)=creq.\overline{ers.s}.(i-1)$, with for the resetting: reset $(i)=\overline{creq}.ers.\overline{s(i+1)}$. These conditions can be simply monitored by means of logic AND-function gates per logic means per section. A drawback of this choice, however, consists in that the buffer is not self-stabilizing. An incorrect status s(i' for example, caused by a fault, may give rise to a permanent fault situation.

A second possibility is: $set(i)=creq.\overline{ers}.\bigwedge_{j=0}^{i-1} s(j)$ and for reset $(i)=\overline{creq}.ers.\bigwedge_{j=i+1}^{n-1} s(j)$. These conditions can again be simply monitored by means of AND-function gates per logic means. For the function $\bigwedge$ see above sub a). This choice ensures stability in the buffer: an error in a status s(i) does not give rise to a permanent error situation. The error disappears in the course of time. However, generally a data loss will occur. A third possibility, where the data loss is minimised, is the preferred solution, consisting in that $set(i)=creq.\overline{ers}.s.(i-1)$, where $reset(i)=\overline{creq}.ers.\bigwedge_{j=i+1}^{n-1} \overline{s(j)}$. The realisation thereof can again be simply realised by the logic means. This is shown in Figure 5 by way of the AND-function gates E1 and E2. The gate E1 monitors the condition $\overline{creq}.ers.\ s(i-1)$ and the gate E2 monitores the condition $creq.\overline{ers}.\bigwedge_{j=i+1}^{n-1} \overline{s(j)}$ whereby the input S of FFi is activated if these conditions are satisfied. In the former case, s(i)=1, whilst in the latter case s(i)=1.

It is to be noted that the gates P1 and P2 shown in Figure 2 can also be assumed to be included in each of the logic means LM(i), signal lines "ers" and "creq" extending along all logic means per section instead of $creq.\overline{ers}$. and $\overline{creq}.ers$, see Figure 6. For proper operation, the condition for set(i) can then be monitored in the gate E'1 by way of "creq" and the inverted "ers" and the status s(i-1). Similarly, the condition for reset (i) can then be monitored in the gate E2 by way of the inverted "creq" and "ers" and the condition $\bigwedge_{j=i+1}^{n-1} \overline{s(j)}$.

The arrangement of Figure 5 is universal for all logic means LM(0),. . . LM(n-1). For LM(0) the input status s(i-1) will not be present; however, this input will require a permanent logic value "1". Similarly LM(n-1) for the input with $\bigwedge_{j=i+1}^{n-1} \overline{s(j)}$ will have the permanent value "1". In the case of extension of the buffer, these inputs can be included in the normal signals paths again as desired in order to enable coupling to a preceding or subsequent buffer. Thus, an extremely simple extension method is realised.

As regards the simple possibility of extending the buff r, it is also to be noted that this extension, notably when solid state integrated buffers are concerned, does not necessarily mean that all signal lines (to a next or preceding buffer) must be extended. It is sufficient to interconnect a "creq" signal input from a next buffer to an SFI signal output of a preceding buffer, and to connect a signal input "ers" of a preceding buffer to a "cack" signal output of a subsequent buffer. The OUTB, obviously, is connected to the IN lines of successive buffers. It is to be noted that in this case the delay time is increased; per connection additional buffer, the delay time increases by one unit (the minimum delay time through a buffer is taken as one unit). According to this solution, howeverm a buffer IC does not require an excessive number of input and output terminals.

CLAIMS

1. A data buffer memory of the "first-in, first-out" type, comprising a plurality of registers arranged in a series array, an input which is connected to a first of the registers and *via* which data to be written are introduced, an output bus coupled to all the registers *via* which bus data are read from the buffer, and logic means for ensuring that data written into the buffer is advanced register by register from the first of the registers and for assigning the register from which data is to be read-out.

2. A data buffer memory as claimed in Claim 1, wherein there are *n* sections (0,1 . . .,n-1) each section containing one of the registers, and the logic means is arranged so that in operation the following signals can be generated:
a) $selout(i) = s(i).\bigwedge_{j=i+1}^{n-1} \overline{s(j)}$ which. if this condition is satisfied, represents the signal whereby the register (i) being the first filled register, viewed from the last (nth) section of the buffer, can be selected and hence connected to the output bus; s(i) indicating that the register (i) contains data and $\bigwedge_{j=i+1}^{n-1} \overline{s(j)}$ indicating that the registers downstream of the register (i) are empty; and
b) $sh = \overline{s(n-1)}.creq.$ which if this condition is satisfied, is the shift signal for shifting the entire content of the buffer over one section in response to the appearance of an externally applied request signal (creq) indicating that data are applied to the input, this request being granted if at least the register of the nth section of the buffer is empty; this is indicated by a status signal $\overline{s(n-1)}=1$ of the latter section.

3. A data buffer memory as claimed in Claim 2, wherein the logic means in operation is capable of generating a signal s(i):=1 which, the condition $creq.\overline{ers}.s.(i-1)$ is satisfied, is the status signal indicating the filling of a register (i) as a result of a request signal (creq) whereby the first register (0) is filled with data applied to the input from outside the buffer or a register (i) i ≠ 0 is filled with data in reaction to a shift signal (sh) whilst none of these registers has been emptied at the same time *via* the output bus, denoted by an externally applied acknowledge signal (ers) and at least the preceding register (i-1) with the exception of i=0 has been filled; the status signal s(i):=0 indicates that the register (i) has been emptied *via* the output bus and has not been filled again at the same time, at least the next register (i+1) being empty (written as $\overline{s(i)}=\overline{creq}.ers.s.\overline{(i+1)}$).

4. A data buff r memory as claimed in Claim 2, wherein for the formation of a status signal s(i):=1 with the logic m ans, $creq.\overline{ers}.\bigwedge_{j=0}^{i-1} s(j)$ has be n

satisfied, which means that when a write request appears on the input, (register (0)) or the shifting in a register (i) respectively and the fact that the register (i+1) is not read at the same time (ers), all registers preceding the register (i) have been filled and that the status signal s(i):=0 if the condition $\overline{creq}.ers.\prod_{j=i+1}^{n-1}\overline{s(j)}$ has been satisfied; this occurs in the absence of a request for writing (creq) and when register (i) is being emptied (ers) and subject to the condition that all subsequent registers are empty.

5. A data buffer memory as claimed in Claim 2, wherein in operation a status signal (s(i):=1) applies if the condition $\overline{creq}.ers.s(i-1)$ is satisfied in the logic means which in the case upon appearance of a request for writing (creq) and non-simultaneous reading ($\overline{ers}$) of the preceding filled register (i-1) (except for i=0); and the status signal s(i):=0 it applies when the condition $\overline{creq}.ers.\prod_{j=i+1}^{n-1}\overline{s(j)}$ is satisfied, which is the case in the absence of a request for writing (creq) and during the emptying (ers) of the register (i) and subject to the condition that all subsequent registers are empty.

6. A data buffer memory as claimed in any one of the preceding Claims, wherein the logic means has an output on which a status signal SFI=s(0) appears in order to indicate that data are present in at least one of the registers of the buffer.

7. A data buffer memory as claimed in any one of the preceding Claims, constructed in solid state integrated technique.

8. A data buffer memory as claimed in Claim 7, wherein each of the registers and its associated logic means form a solid state integrated circuit which comprises a section of the buffer.

9. A data buffer memory as claimed in Claim 7, wherein the buffer consists of at least one group of registers and at least one group of logic means, said groups being respective solid state integrated circuits.

10. A data buffer memory constructed and arranged to operate substantially as hereinbefore described with reference to and as shown in the accompany drawings.

Printed for Her Majesty's Stationery Office by Croydon Printing Company Limited, Croydon Surrey, 1979.
Published by the Patent Office, 25 Southampton Buildings, London, WC2A 1AY, from which copies may be obtained.